“As Simple as Possible but not Simpler”: Towards an Ontology Model Canvas

Giancarlo GUIZZARDI[a,c], Tiago Prince SALES[b,c]
[a]*Conceptual & Cognitive Modeling Research Group (CORE), Free University of Bozen-Bolzano, Bolzano, Italy*
[b]*University of Trento, Italy*
[c]*Ontology & Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo (UFES), Vitória, Brazil*

Abstract. Over the years, there has been an increasing adoption of ontology-driven conceptual models to represent the conceptual structure of critical domains in reality. Given the complexity of this task, there has been a growing demand for the development of proper engineering tools for supporting the design of these models. Despite a number of advances in this area, there is still a shortage of tools directed at novice and non-technical users that can, at the same time, address two competing requirements, namely: maintain modeling expressivity by being able to represent true ontological distinctions, while remaining intuitive and easy to learn by this class of users. In this article, we sketch a proposal in this direction by introducing the idea of an *Ontology Model Canvas*.

Keywords. UFO, OntoUML, Ontology-driven Conceptual Modeling

1. Introduction

In recent years, there has been an increasing interest in the application of ontologies in conceptual modeling, including the use of foundational ontological theories to improve the theory and practice of this discipline [1,2]. In these scenarios, ontological theories can play a fundamental role in improving the quality of enterprise-wide conceptual models such that they can properly serve as artifacts supporting communication, problem-solving, meaning negotiation and, chiefly, semantic interoperability in its various manifestations.

Given the increasing complexity of Ontology-driven Conceptual Modeling, there is an urging need for developing theoretically well-founded methodological and computational tools for this discipline [1,3]. In order to contribute to this goal, we have, over the years, developed a number of such tools, including: a UML-based visual Ontology-driven Conceptual Modeling language (dubbed *OntoUML*) [4,5], catalogues of Ontology Design Patterns and Anti-Patterns [6-8], Ontology Pattern Languages [6], as well as tools for model construction, verbalization, verification and validation via visual simulation [1,8][1]. These tools have been developed with the theoretical support of the Unified Foundational Ontology (UFO) [4,5], an axiomatic formal theory based

[1]This paper is a companion to the keynote talk given by the first author at the 3rd International Joint Ontology Workshop (JOWO 2017), in Bozen-Bolzano, Italy. The keynote speech was generally devoted to discussing these tools that have been developed with the goal of combining, on one hand, expressivity and ontological rigor and, on the other hand, the need to shield the user from the complexities of the ontology engineering process. This paper, in contrast, focuses on one specific tool of this kind.

on theories from Formal Ontology, Philosophical Logics, Cognitive Psychology and Linguistics.

The results of these efforts have been noticed in academic, industrial and governmental settings. For example, OntoUML has been successfully employed in several industrial projects in different domains, such as petroleum and gas, digital journalism, complex digital media management, off-shore software engineering, telecommunications, retail product recommendation, and government [5]. Moreover, a recent study shows that UFO is the second-most used foundational ontology in conceptual modeling and the one with the fastest adoption rate [2], and that OntoUML is among the most used languages in Ontology-driven Conceptual Modeling.

Despite the theoretical and practical contributions of these efforts, these tools have not been particularly developed with novice and non-technical users in mind. Especially for these classes of users, we need tools that are able to, on one hand, deal with the inherent complexity of the modeling problems at hand, and on the other hand, shield these users as much as possible from this inherent complexity. In this paper, we advance a proposal in this direction by conceiving an ontology-based modeling representation mechanism that, despite having the expressivity of a large set of ontological distinctions put forth by UFO, could in theory be easily learned and employed by novice and non-technical users. It is important to highlight that we are here specifically trying to avoid the route taken by modeling approaches that are oblivious to true ontological distinctions and, hence, that are in our opinion too simplistic for supporting the development of Reference Ontologies [3]. To put it baldly, we are aiming at conceiving an ontology-modeling representation mechanism that should be, paraphrasing Albert Einstein, "as simple as possible, but not simpler".

In this proposal, we took inspiration on a significant success case in the area of Business Modeling, namely, the Business Model Canvas (BMC). The BMC, proposed in [10] is a modeling tool that is recurrently reported by practitioners and students as being both useful and easy to learn. In addition to drawing from this inspiration, we base this preliminary proposal of an *Ontology Model Canvas (OMC)* on empirical data of how UFO categories are actually used in practice in the construction of conceptual models in a variety of domains and settings. This data has been collected from the OntoUML model repository[2] [8], a repository of 54 models of different sizes (e.g., ranging from a dozen to thousands of types), representing different domains (e.g., Biodiversity, Telecommunications, Services) and created in different environments (e.g., ranging from single-authored models created in academic environments to models created by teams in practical settings over the course of years). The accumulated practical experience in the use of OntoUML as well as the analysis of the models in this repository allowed us to identify, for instance, which constructs represent to the core elements of the language that are most frequently used in practice. It also allowed us to identify which should be the default interpretation of some of the language constructs. Finally, as a third influence to the ideas sketched here, we took into account a few important insights on visual cognition and visual concrete syntax design elaborated in [11,12].

The remainder of this paper is organized as follows. In the next section, we present a whirlwind introduction to those categories of UFO that are germane to the purposes of this article. In section 3, which is the core of this article, we present our preliminary proposal of an Ontology Modeling Canvas. Section 4 presents some additional

[2] http://www.menthor.net/model-repository.html

examples that serve to illustrate the modeling expressivity that can be achieved with such an approach. Finally, section 5, presents some final considerations of the paper.

2. A Whirlwind Introduction to a Fragment of the UFO

In the sequel, we briefly explain a selected subset of the ontological distinctions put forth by the Unified Foundational Ontology (UFO). For an in depth discussion, philosophical justifications, formal characterization and empirical support for these categories one should refer to [4,13].

Take a domain in reality restricted to endurants [4] (as opposed to events, occurrents). Central to this domain we will have a number of **Object Kinds**, i.e., the genuine fundamental types of objects that exist in this domain. The term "kind" is meant here in a strong technical sense, i.e., by a kind we mean a type capturing essential properties of the things it classifies. In other words, the objects classified by that kind could not possibly exist without being of that specific kind. Kinds tessellate the possible space of objects in that domain, i.e., all objects belong to exactly *one kind* and do so *necessarily*. Typical examples of kinds include Person, Organization and Dog. We can, however, have other static subdivisions (or subtypes) of a kind. These are naturally termed **Object Subkinds**. As an example, the kind Person can be specialized in the subkinds Man and Woman.

Object kinds and subkinds represent essential properties of objects (they are also termed rigid or static types [4]). We have, however, types that represent contingent or accidental properties of objects (termed anti-rigid types [4]). These include **Phases** (for example, in the way that being a living person captures a cluster of contingent properties of a person, or in the way that being a puppy captures a cluster of contingent properties of a dog) and **Roles** (for example, in the way that being a husband captures a cluster of contingent properties of a man). The difference between the contingent properties represented by a phase and a role is the following: phases represent properties that are intrinsic to entities (e.g., being a puppy is being a dog that is in a particular developmental phase; being a living person is being a person who has the intrinsic property of being alive); roles, in contrast, represent properties that entities have in a relational context, i.e., contingent relational properties (e.g., being a husband is to bear a number of commitments and claims towards a spouse in the scope of a marriage relationship; being a student is to bear a number of properties in the scope of an enrollment relationship with an educational institution).

Kinds, Subkinds, Phases and Roles are categories of **Object Sortals**. In the philosophical literature, a sortal is a type that provides a uniform principle of identity, persistence and individuation for its instances [4]. To put it simply, a sortal is either a kind (e.g., Person) or a specialization of a kind (e.g., Student, Teenager, Woman), i.e., it is either a type representing the essence of what things are or a sub-classification applied to the entities that "have that same type of essence".

A **Relationship Kind** represents clusters of relational properties that "hang together" by a nexus (provided by that relationship kind). In other words, relationships (e.g., marriages, enrollments, employments, presidential mandates, citizenships) are full-fledged endurants, i.e., entities that endure in time bearing their own essential and accidental properties and, hence, first-class entities that can change in a qualitative manner while maintaining their identity [13].

Objects participate in relationships playing certain "roles". For instance, people play the role of spouse in a marriage relationship; a person plays the role of president in a presidential mandate. Spouse and President (but also typically student, teacher, pet) are examples of what we technically term a role in UFO, i.e., a relational contingent *sortal* (since these roles can only be played by entities of a unique given kind). There are, however, relational and contingent role-like types that can be played by entities of multiple kinds. An example is the "role" Customer (which can be played by both people and organizations). We call these role-like types that classify entities of multiple kinds **Rolemixins**. In general, types that represent properties shared by entities of multiple kinds are termed Non-Sortals. In UFO, besides rolemixins, we have two other types of non-sortals, namely **Categories** and **Mixins**. Categories represent necessary properties that are shared by entities of multiple kinds (e.g., the category Physical Object represent properties of all kinds of entities that have masses, spatial extensions, etc.). In contrast, mixins represent shared properties that are necessary to some of its instances but accidental to others (e.g., the mixin Red Object can be thought as representing properties that are necessary to entities of certain kinds – for instance, rubies, while being accidental to entities of other kinds – for instance, apples). Categories and mixins are, in contrast to rolemixins, considered as Relationally Independent Non-Sortals.

3. Towards a Well-Grounded Ontology Model Canvas

Figure 1 illustrates the different regions comprising our tentative proposal for an Ontology Model Canvas (OMC). These regions are elaborated in subsections from 2.1 to 2.5 below. Loosely speaking, the central part of the canvas is divided in four regions. The two regions to the left (i.e., regions corresponding to *categories*, and *kinds/subkind*s) should be read in as "the ways things necessarily are" (according to a given conceptualization of the domain); the two regions on the right (i.e., regions corresponding *phases* and *roles*) should be read as the "the ways things can possibly be". The region corresponding to Mixins "crosscut" these two macro-regions, given that mixins capture semi-rigid types, i.e., types are *necessarily* instantiated by some instances and contingently (*possibly*) instantiated by other instances. In the bottom part of the canvas, we represent relationship kinds and subkinds.

The regions partitioning this canvas reflect the UFO categories of types that are the most recurrent in models according to the OntoUML repository [14]: role (1655 occurrences or 27,6% of all occurrences of types), subkind (1169 occurrences or 19,5%), relationship type[3] (with 1089 occurrences or 18,2%), kind (837 occurrences or 14%), rolemixin (278 occurrences or 4,6%), categories (217 occurrences or 3,6%) and mixins (54 occurrences or 1%). For the sake of completeness of the UFO typology of object types, and also because taxonomic constraints can require the explicit representation of phases, we have also decided to include a phase region in the canvas. Still, with 207 occurrences, phases appear in the models of the repository more frequently than the remaining object types, namely, *collectives* and *quantities* [4]. The OMC only allows the representation of two relations, namely generalization and mediation. The former is a relation inherited from UML, which is used to construct hierarchies of types and is by far the most used relation in these models. The latter is a

[3] Termed in [14] a relator type.

special type of OntoUML relation, which is used to connect relationship types (relator types) and their bearers. In the OntoUML repository, mediations were used 1718 times.

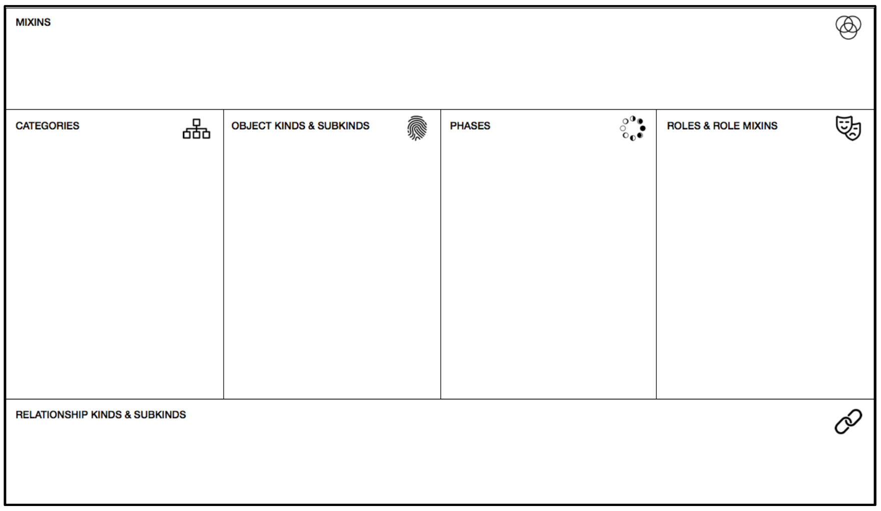


Figure 1. The Topology of the Ontology Model Canvas (OMC)

3.1. Object Kinds and Subkinds

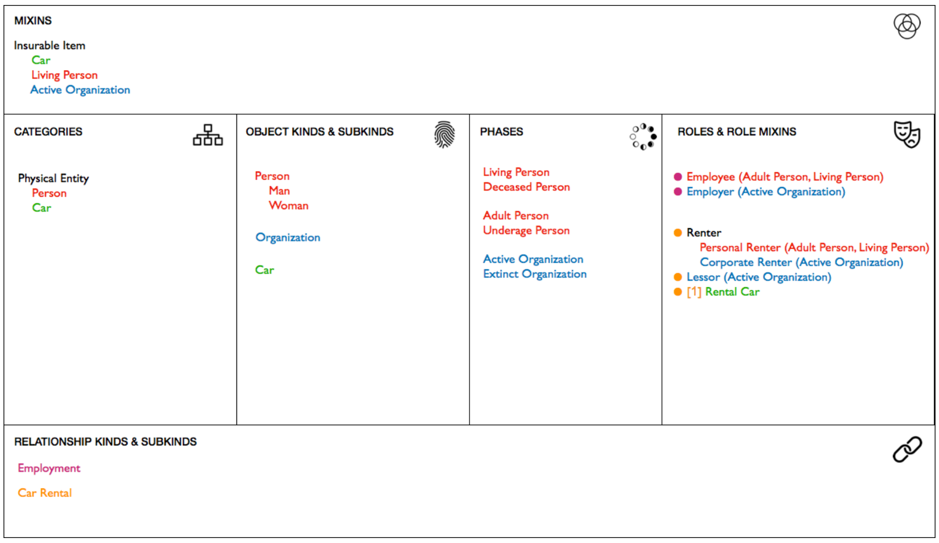
In this region of the canvas, we have all the rigid object sortals of the domain at hand. As previously mentioned, **kinds** tessellate the universe of objects in the domain, i.e., kinds are exhaustive (all objects belong to a kind) and all kinds are mutually disjoint (no object belongs to more than one kind). In this region, different kinds are represented in a different color. The color variable is used here to trace the line of inheritance of the *principle of identity* provided by a kind [4]. In the model depicted in Figure 2 below, we have three kinds, namely, Person, Organization and Car, each of which is represented by a different color. The canvas makes it explicit (and, thus, efficient to answer in visual queries) the question of *"what fundamental KINDS of things exist in this domain?"*

As figure 2 shows, we use indentation (tabs) to represent a subtyping relation. For example, Man and Woman are subtypes of Person. In fact, these are **subkinds** of Person, i.e., rigid subtypes of a kind that share the *principle of identity* provided by that kind (hence the representation of subkinds using the same color of their respective kinds). We assume that unless explicitly specified, all subtypes of a type defined in an indentation scope (all those subkinds in subsequent lines that share the same indentation) form a *partition* of that given type (i.e., they form a disjoint and complete *generalization set* [4][4]). Notice that having (by default) specializations to be defined in a partition is generally considered a good practice by the Information Science, Conceptual Modeling and Formal Ontology communities [15].

In order to visually differentiate the multiple regions of the canvas, we make use of *signs* (*indexes*, actually, in the semiotic sense [4]) in the top-right corner of each region to indicate some of the prominent meta-properties of the corresponding represented

[4]Henceforth, when using the term 'partition', we always use it in this technical sense, i.e., as a disjoint and complete tessellation of the space of instances at hand.

object category. We believe that this helps to address the requirements of *perceptual discriminability* (roughly, how easy it is to visually discriminate a construct from other constructs of the language) and *perceptual immediacy* (how easy it is for a user of the language to infer the real-world category represented by that construct) discussed in [11,12]. Since kinds are the types that provide a principle of identity for their instances, we use a "fingerprint" sign (as a stereotypical proxy for identity) to symbolize this connection. As previously mentioned, subkinds are rigid subtypes of kinds. The use indentation to represent the subtyping relation explores the property of *directness* in visual representation systems affording pragmatically efficient cognitive operations called *inferential free-rides* [12]. This is because the visual relational primitive (*to-the-right-of*) used to represent the ontological relations of subtyping has the same formal meta-properties of the relation itself, i.e., it is a partial order relation. Finally, the use of colors to differentiate kinds also speaks to perceptual discriminability and, even, *perceptual popout*, given that colors, being processed by a parallel perceptual system [16] are extremely efficient variables for visual queries.


MIXINS

Insurable Item
Car
Living Person
Active Organization

CATEGORIES

Physical Entity
Person
Car

OBJECT KINDS & SUBKINDS

Person
Man
Woman

Organization

Car

PHASES

Living Person
Deceased Person

Adult Person
Underage Person

Active Organization
Extinct Organization

ROLES & ROLE MIXINS

● Employee (Adult Person, Living Person)
● Employer (Active Organization)

● Renter
Personal Renter (Adult Person, Living Person)
Corporate Renter (Active Organization)
● Lessor (Active Organization)
● [1] Rental Car

RELATIONSHIP KINDS & SUBKINDS

Employment

Car Rental

Figure 2. Example of an instance model in the Ontology Model Canvas

3.2. Relationship Kinds and Subkinds

After considering the kinds of entities that exist in a given domain, one can move on to consider what kinds of **relationships** can hold between the entities in that domain. Here, once more, the different relationship kinds tessellate the space of possible relationships and, again, we use different colors to trace the identity of different relationship kinds and make use of indentation (tabs) and consistency of color to represent subkinds of a relationship kind. In figure 2, we have two relationship kinds, namely, Employment and Car Rental. One should remember that in the theory of relationships in UFO, relationships are not n-uples but full-fledged endurants [13]. So, for instance, a particular employment instance (e.g., between John and the United Nations) is a complex endurant comprising a number of relational properties.

We use a "double ring/chain" sign to represent the bond or tie between the entities a relationship connects. Here as well, this choice is meant to address the requirement of *perceptual immediacy*. One should notice that relationship kinds are also rigid types (e.g., the particular employment of John in the United Nations is necessarily an employment relationship) that provide a uniform principle of identity for their instances. However, the use of the "chain" sign here is meant to highlight the primary ontological status of relationships as being existentially dependent on the entities they bind (mediate, relate).

3.3. Roles and Rolemixins

Both **roles** and **rolemixins** are generically relationally dependent types [4], i.e., entities can only instantiate these types in the scope of a relationship to another entity(ies) of a given type (or types). In the inverse direction, relationships induce certain (relational) role(mixin)s and, hence, when entities are bound by a relationship, they do so under certain role(mixin)s (i.e., they are classified under certain role(mixin)s). The difference between roles and rolemixins is that, whilst the former is a **sortal** (i.e., all its instances are of the same kind), the latter is a **non-sortal** (i.e., it is a *dispersive type* [4], classifying entities of multiple kinds).

In this region, we make use of the color scheme of types to indicate which kinds of entities play a particular role. In other words, as formally proved in [4], every role specializes a unique kind. In the case of rolemixins (actually, for non-sortals, generally), we reserve the black color to represent that the rolemixin itself does not carry a uniform principle of identity to its instances, i.e., that the instances of a rolemixins are not committed to being of a particular unique kind. Rolemixins are, therefore, necessarily specialized into roles forming a subtyping partition (represented using the indentation scheme previously discussed). In Figure 2, we have an example of the rolemixin Renter, which is specialized into the roles Personal Renter (played by entities of the kind Person, as indicated by the red color) and the role Corporate Renter (played by entities of the kind Organization, as indicated by the blue color). Still in this figure, we can immediately see the roles played by Person (Employee and Personal Renter), by Organizations (Employer, Lessor and Corporate Renter) and by Cars (Rental Car).

Since role(mixin)s are always defined in the scope of a relationship type, we use a circle with the corresponding color of a relationship kind to bind a role(mixin) to that relationship kind. For example, in Figure 2, we have that an Employment relationship kind defines two roles, namely, Employer and Employee, whilst the Car Rental relationship kind defines three role(mixin)s, namely, Lessor, Renter and Rental Car (each of which is directly visually mapped to the kinds of entities that can play those roles). Once more, we use visual variables to afford efficient parallel processing of visual queries (e.g., inspecting a canvas for the role involved in a relationship and the kinds of entities that can play them). The adjacency relation between a circle filled with the color of relationship kind and a role(mixin) in this partition represents the type-level relation of *mediation* [4] between the former and the latter. So, for example, in Figure 2, we have that Employment *mediates* Employer and Employee, which translates to the instance-level constraint that every instance of Employment is existentially dependent on particular instances of Employer and Employee, but also to the instance-level constraint that, for example, in order to play the roles of an Employee, a Person has to be mediated by a particular instance of Employment.

As previously discussed, role(mixin)s are relationally dependent types. As consequence, the minimum cardinality constraint from the role to the relationship type is at least one (and typically exactly 1) and the maximum cardinality constraint is typically *many* (* in (Onto)UML). For this reason, we adopt these as the default interpretation of the cardinality constraint in the mediation relation between a role(mixin) and the corresponding relationship type (for example, in figure 2, we should read that "every Employer is mediated by (participate in) one-to-many employments"). Analogously, since relationships are existentially dependent entities [4,13], the minimum cardinality constraint from the relationship kind to a role(mixin) must be higher than one and, typically, exactly 1. The maximum cardinality constraint of this relation in this direction is also typically of exactly 1[5]. For this reason, we adopt as *exactly one* (1..1) the default interpretation the cardinality constraint in the mediation relation from a relationship type to a role(mixin) (for example, in figure 2, we would read that every Employment mediates exactly one Employer).

Whenever cardinality constraints in this mediation relation deviates from these default values, we represent the deviating constraint the same identity color scheme to represent the direction of the constraint plus brackets ("[ ]") to represent the values imposed by the constraint itself. For instance, in figure 2, we represent that a Rental Car can participate in exactly one Car Rental[6]. Since the cardinality constraint, in this case, refers to the association end connected to the relationship kind (Car Rental), we use the color of the latter also for the constraint. Following the UML notation, we have cardinality constraints represented in the form [a..b] (where b ≥ a) and that the most typical combinations of value are [1..1] ([1], for short) [1..n] ([+], for short). Note that [0..n] ([*], for short) is not an option here, given the aforementioned constraints on generic relational dependence on one side and, existential dependence, on the other.

Once more, we emphasize these cardinality constraints are only represented when they deviate from the default values. Although the instances of a role are always of a unique kind (hence, we have the role name represented in the color of the kind at hand), these roles at times do not subtype these kinds directly but a particular set of subtypes of that kind. In figure 2, for instance, an Employer must be not only an Organization but, more specifically, an Active Organization (and the same for Corporate Renter); an Employee must not only be a Person but an Adult and a Living Person (and the same for Personal Renter). We use parenthesis to the right of type name to represent the direct supertype of that type (within the specialization taxonomy of a given kind). In these examples from Figure 2, these direct supertypes are not subkinds of the kind of entities that play that role, but *phases* of that kind (see section 2.4).

Finally, we use a "Greek theater masks" sign to mark the role(mixin) region. This is meant to evoke the idea of roles as context dependent and contingent (as roles in a play are). Again, this visual representation choice addresses *perceptual immediacy* and *discriminability*.

3.4. Phases

Like roles and rolemixins, phases are anti-rigid (dynamic classification) types [4]. Moreover, like roles, phases are sortals that specialize a unique kind that provide a uniform principle of identity for their instances. Once more, we make use of the color

[5] Again, based on our experience, anecdotal evidence and analysis of models in the OntoUML repository.

[6] This is, of course, arguable as a modeling choice, especially if one takes a historical view, in which a rental car can be associated to several rentals in different periods of time. This modeling choice is used here simply as an example to demonstrate a deviation of the default cardinality and how it should be represented. In this example, had we had the cardinality constraint deviation stating that a Car Rental could bind one or more rental cars, we would have the constraint [+] in green also represented next to the Rental Car role.

variable to represent of which kind a particular type is a *phase of*. However, unlike roles, phases are always defined in disjoint and complete generalization sets (termed *phase partitions*) [4,6]. We here represent phase partitions by using sets of phase terms under the same color, grouped together and perceptually apart from other phase groups (partitions). We use a blank line to separate different phase partitions. In Figure 2, we have three *phase partitions*: Adult Person and Underage Person that define phases of a Person; Living Person and Deceased Person that define phases of a Person; Active Organization and Extinct Organization as phases of Organization. Given the semantics of phases and phase partitions, we have that, for example, all persons are either an Adult Person or an Underage Person; no person is both Adult Person or an Underage Person; every person that instantiates Adult Person in a situation instantiates Underage Person in a counterfactual situation, and vice-versa.

Finally, we make use of a sign showing "phases of the moon" to mark this region. This is meant to evoke the idea that being in a phase is a contingent property for entities of that kind but also the idea that phases are associated to other complementary phases (forming a phase partition).

3.5. Categories and Mixins

Non-sortals are types that classify entities of multiple kinds. These include the so-called semi-rigid non-sortals called simply **Mixins** describing properties that are essential to some instances and accidental to others and rigid non-sortals called **Categories**. An example of the latter can be found in Figure 2: Physical Entity classifies both entities of the kind Person and entities of the kind Car. In this figure, we also have an example of the former, namely, Insurable Item, which can be thought to be essential for Cars but accidental for People (only living people are insurable) as well as for Organizations (only Active Organizations are insurable).

Contrary to subtype partitions in other regions, in the case of these partitions in the non-sortal region, we assume that, by default, these are disjoint (since all kinds are mutually disjoint) but not complete (e.g., there are of course other Physical Entities that are neither cars nor persons).

We believe that representing relationally independent non-sortals separately from rolemixins is justified by a number of aspects that we have learned from our experience and other people's experience in using OntoUML. The first of these reasons has to do with relational dependence. Because of this meta-property, rolemixins only appear in models connected to relationships and this justifies their modeling in the *relationally dependent types* region of the canvas. The second reason (related to the first) is that, methodologically speaking, rolemixins appear in models in a modeling stage that is much closer to how roles are modeled than to how other non-sortals are modeled. Basically, rolemixins typically firstly appear in models considered as roles, i.e., dynamic types connected to a relational context. Then, in a second stage, one realizes that these alleged "roles" could not be easily connected to a taxonomy of kinds. This is exactly because they can be played by entities of multiple kinds, i.e., they are in fact rolemixins. This modeling problem is so common that it is actually described as a *problem pattern* in [4]. The other types of non-sortals, on the other hand, typically appear in models as a result of model refactoring operations. For example, the type Physical Entity (or Insured item) is typically included in a model as the result of an operation of abstracting and representing in a single supertype a property (spatial extension, insurance) that is shared by entities of multiple kinds. Finally, unlike in the case rolemixins, subtype partitions of relationally-independent non-sortals are seldom complete. To put it simply, rolemixins tend to be conceived in a top-down manner, i.e., firstly one thinks of the rolemixin (again, firstly considered as a "role") and then, after realizing that it cannot carry a uniform principle of identity for its instances, one goes after finding out the multiple "hidden" roles instantiated by the entities of the multiple kinds that instantiate that rolemixin; the other types of non-sortals, in contrast, are typically conceived in a botton-up manner as a model refactoring and complexity management mechanism.

The sign we propose for the mixin region is a Venn diagram, in order to convey the notion that mixins aggregate intersections of multiple kinds. For the category region, we chose a hierarchy symbol, which conveys the abstraction/refactoring mechanism supplied by categories.

We show below an OntoUML model that is equivalent to the model of figure 2. This model can be automatically generated from the model of figure 2. As such, this OntoUML can clearly be taken as an alternative visualization of the original instantiation of the corresponding OMC. However, due to use of a standardized topology for the regions of the canvas, use of colors and default readings (e.g., cardinality constraints and generalization sets), we believe that the canvas can provide for an interesting alternative concrete representation for novice or non-technical users.

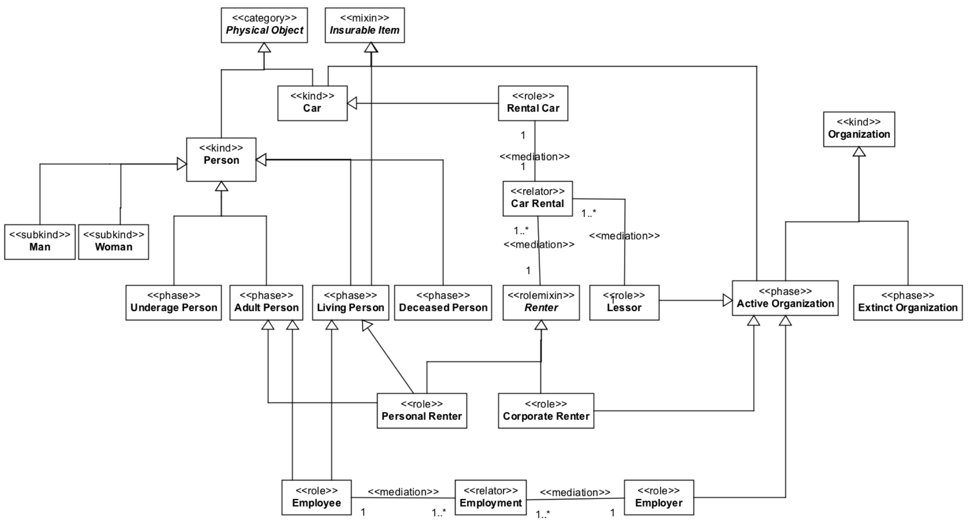


Figure 3. OntoUML model equivalent to the OMC instantiation in Fig.2.

4. A More Complex Example

As a second illustrative example of an instantiation of the Ontology Model Canvas, we propose the model of figure 4. This second example revisits the more complex modeling case presented in [7][7]. The model is used there to represent relations between certain ontology design patterns and contains cases of multiple subtying, overlapping generalization sets and association subtyping.

[7]The only different between the model of Fig.4 and the one in [7] is that the former makes explicit the relationship behind the material relation connecting Project and Project Party in the latter.

In this case, we have three kinds of objects (Persons, Organizations and Projects) in this domain and three kinds of relationships involving them (Enrollment, Employment and Project Assignment). An Enrollment is a relationship kind mediating Students (also of the kind Person) and Universities (of the Organization subkind University).

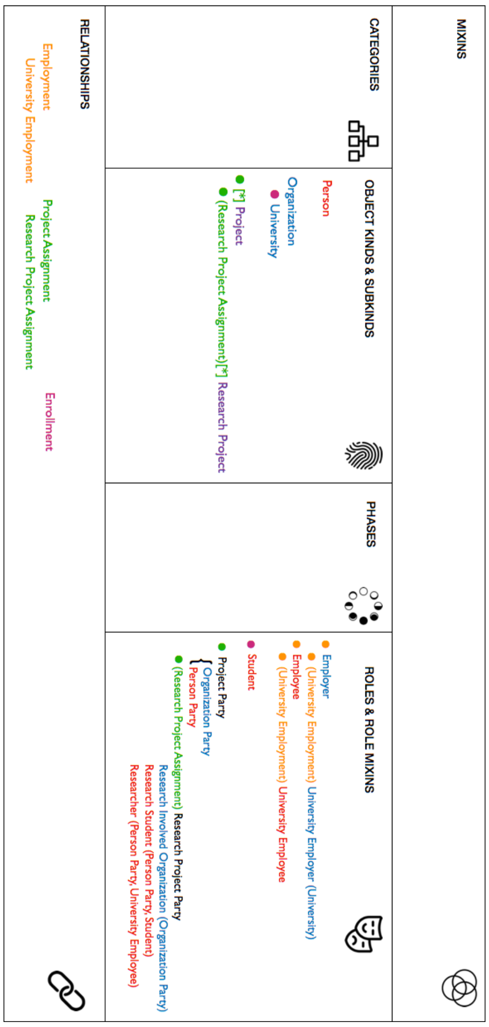

MIXINS
CATEGORIES
OBJECT KINDS & SUBKINDS
Person
Organization
● University
● [*] Project
● (Research Project Assignment)[*] Research Project
PHASES
ROLES & ROLE MIXINS
● Employer
● (University Employment) University Employer (University)
● Employee
● (University Employment) University Employee
● Student
● Project Party
Organization Party
Person Party
● (Research Project Assignment) Research Project Party
Research Involved Organization (Organization Party)
Research Student (Person Party, Student)
Researcher (Person Party, University Employee)
RELATIONSHIPS
Employment
University Employment
Project Assignment
Research Project Assignment
Enrollment

Figure 4. Using the OMC to represent an OntoUML model from [7]

In Figure 4, we can also see that Employment is a relationship binding Employers (which are Organizations) and Employees (who are of the kind Person). The model also shows that we have a particular subkind of Employment that is a University Employment. The role University Employee is a subtype of Employee, like for all Employees, this is a role played by people. Moreover, like all Employees, University Employees are *mediated by* Employment relations. However, University Employees are mediated by a particular subkind of Employment, namely, University Employment. Since a University Employee is an Employee, the default interpretation of the mediation relation between Employment and University Employee of a sub-relation of the mediation relation between Employment and Employee[8]. Previously, we have used parenthesis to indicate a direct subtyping constraint (e.g., Personal Renter is an Adult, Living Person). We here also use parenthesis to capture a subtyping constraint on the specialization of this mediation relation, i.e., the model captures that University Employees are mediated not by the general relation with Employment but by a specialization of that relation with a subkind of Employment, namely, University Employment. Analogously, the model in the canvas shows that a University Employer: is an Organization of a particular type, namely, of the subkind University; that it is mediated by a particular subkind of Employment, namely, University Employment.

Finally, we have that Project Assignment is a relationship involving a Project and a Project Party. Projects can exist without being associated to any Project Assignment (the *zero-to-many* [*] cardinality constraint on the association end connected to the latter that deviates from the default constraints for mediation relations) and Research Projects (a subkind of Project) is (again, optionally) mediated by a particular subkind of assignment (Project Assignments). Project Party is a rolemixin and, hence, it should be specialized into a partition of roles that assign different kinds to its instances. Here, we have that a Project Party can be either a Person Party (i.e., of the kind Person) or an Organization Party (i.e., of the kind Organization). As we previously mentioned, the default interpretation of subtypes that appear in a sequence of indentation is that they form a partition. For this reason, we have to explicitly represent when this is not case in a model. In this example, it is not the case that all the subtypes of Project Party are disjoint, in fact, (a) Research Student and (b) Researcher are Person Parties, and (c) Research Involved Organizations are Organization Parties in a project. The types (a-c) are, in turn, subtypes of Research Project Party (a rolemixin), which are Project Parties that are constrained to be associated in that role to a subkind of Project Assignment (Research Project Assignment). Finally, we would like to still represent that a subset of the subtypes of Project Party indeed form a partition of that type, namely, that Person Party and Organization Party tessellate the role Project Party. This is explicitly represented in the canvas notation by the use of braces ("{") around the subtypes forming a partition.

5. Final Considerations

In this paper, we elaborate on a first attempt to conceive an ontology modeling representation mechanism targeted at novice and non-technical users. This mechanism should, on one hand, have enough expressivity to represent important ontological distinctions that must be explicit represented in complex domains and, on the other hand, remain intuitive and relatively easy to learn to this class of users.

In order to target the first requirement, we ground our proposal on the Unified Foundational Ontology (UFO) as well as on the accumulated experience of applying the UFO-based conceptual modeling language OntoUML (and its associated methodological and computational tools) in a number of projects in a diversity of

[8] One should notice that here we are simply assuming that there is an inclusion constraint between sub and super-relations, i.e., that the instances of the sub-relation are instances of the super-relation. We are not, however, discriminating if these relations between relations are ones of *subsetting*, *association specialization* or *redefinition* [17]. As shown in [17], the theory of relations put forth by UFO is particularly expressive in identifying and differentiating these cases.

settings and domains; to address the second requirement, we ground the proposal on empirical data of the use of UFO and OntoUML, on some guidelines from the literature of visual concrete syntax analysis and design [11,12], and by taking an explicit inspiration on the Business Model Canvas of Osterwalder and Pigneur [10].

What we presented here is just a first sketch of the idea of an Ontology Model Canvas (OMC). Everything else remains to be done for this representation mechanism, including: extend it to represent other UFO categories (e.g., intrinsic properties, events, parthood), provide a formal definition of abstract syntax and semantics; build proper computational tool support and, perhaps, more importantly, to empirically validate this idea. After all, whether this representation mechanism is indeed easy to learn and employ by users is clearly an empirical question. We shall address all these issues in future work.

Acknowledgements. We thank João Paulo Almeida, Luiz Olavo Bonino, Ricardo Falbo and Renata Guizzardi for feedback on an earlier version of this paper.

References

[1] G. Guizzardi, "Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling", in *Proc. 33rd Intl. Conf. on Conceptual Modeling (ER'14)*, 2014, pp. 13–27.

[2] M. Verdonck, F. Gailly, "Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling", in *Proc. of ER 2016*. LNCS, Vol. 9974, 83-97. Springer (2016).

[3] G. Guizzardi, "Theoretical foundations and engineering tools for building ontologies as reference conceptual models", *Semantic Web Journal*, vol. 1, no. 1-2, pp. 3-10, 2010.

[4] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.

[5] G. Guizzardi et al., "Towards ontological foundation for conceptual modeling: the Unified Foundational Ontology (UFO) story", *Applied Ontology*, vol. 10, no. 3-4, pp. 259-271, 2015.

[6] F. Ruy et al., "From reference ontologies to ontology patterns and back", *Data & Knowledge Engineering*, 2017.

[7] F. Ruy et al., "Ontology engineering by combining ontology patterns", in *Proc. 35th International Conference on Conceptual Modeling (ER 2015)* Stockholm, 2015.

[8] T.P. Sales, G. Guizzardi, "Ontological anti-patterns: empirically uncovered error-prone structures in ontology-driven conceptual models". *Data & Knowledge Engineering*, vol. 99, pp.72–104, 2015.

[9] J. Guerson et al., "OntoUML Lightweight Editor: a model-based environment to build, evaluate and implement reference ontologies", *19th IEEE Enterprise Distributed Object Computing (EDOC), Demo Track*, Adelaide, Australia, 2015.

[10] A. Osterwalder, Y. Pigneur, *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, Inc., 2010.

[11] D.L. Moody, "The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering". *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756-779, 2009.

[12] G. Guizzardi, "Ontology-based evaluation and design of visual conceptual modeling languages", in *Domain engineering*, Iris Reinhartz-Berger, Arnon Sturm, Tony Clark, Jorn Bettin, Sholom Cohen (Editors), Springer-Verlag, 2013.

[13] N. Guarino, G. Guizzardi, ""We need to discuss the relationship": revisiting relationships as modeling constructs", in *Proc. 27th International Conf. on Advanced Information Systems Engineering (CAISE 2015), LNCS 9097*, pp. 279-294.

[14] M.G.S. Teixeira, *An ontology-based process for domain-specific visual language design*, PhD Thesis, Federal University of Espírito Santo, Brazil/Ghent University, Belgium (Double Degree), 2016.

[15] R.J. Wieringa, W. de Jonge, P.A. Spruit, "Using dynamic classes and role classes to model object migration". *Theory and Practice of Object Systems*, vol. 1, no. 1, pp. 61-83, 1995.

[16] R.E. Mayer, R. Moreno, "Nine ways to reduce cognitive load in multimedia learning". *Educational Psychologist*, vol. 38, no. 1, pp. 43-52, 2003.

[17] D. Costal et al., "Formal semantics and ontological analysis for understanding sub-setting, specialization and redefinition of associations in UML", in *Proc. 30th International Conference on Conceptual Modeling (ER 2011)*, Brussels, Belgium, 2011.